AMENDMENTS TO THE CLAIMS

This listing of claims replaces all prior versions, and listings, of claims in the application:

Listing of claims

Claims 1 through 15. **(Cancelled)**

16. (New) In a computing system having one or more computers configured to invoke one or more software services, a semantic-based method of defining and automating service recovery and transactional behavior of a software service, the method comprising:

displaying a graphical presentation of a software service on a graphical user interface, the software service having a service interface definition comprising one or more defined inputs and one or more defined outputs represented the graphical presentation as ports;

defining a context-sensitive attribute of the service interface definition of the software service, the context-sensitive attribute declaring whether the software service is transactional and recovery behavior of the software service upon a termination event, wherein the recovery behavior is at least one of compensate, rollback or commit; and

displaying the context-sensitive attribute in the graphical presentation of the software service as at least one flow port that allows a user to visually route a flow of service invocation to anther service when the termination event occurs.

17. (New) The method of claim 16, wherein the context-sensitive attribute further defines whether to automatically re-invoke an instance of the software service even if a persistent state of execution of the software service indicates that a corresponding service instance was already invoked successfully prior to the termination event.

18. (New) The method of claim 16, wherein the context-sensitive attribute overrides a previously defined attribute in the service interface definition based on a context of usage of the software service within another composite service.

19. (New) The method of claim 16, wherein the context-sensitive attribute extends a connectivity of the at least one flow port.

20. (New) In a computing system having one or more computers configured to invoke one or more composite software services, each composite software service containing one or more embedded software services, a method of creating a context for a composite service, the method comprising:

identifying a composite software service containing one or more embedded software services;

receiving a request to invoke the composite software service;

tracking in an execution graph connectivity and data dependencies among the one or more embedded software services contained within the composite software service including tracking any context-sensitive attributes for the one or more embedded software services, the context-sensitive attributes declaring whether a embedded software service is transactional and recovery behavior of the embedded software service upon a termination event, wherein the recovery behavior is at least one of compensate, rollback or commit;

tracking in an invocation map a context of the composite software service, the context including states of execution of the one or more embedded software services;

associating a unique ID for each of the one or more embedded software services;

invoking one of the one or more embedded software services;

when receiving an indication that invocation of one of the one or more embedded software services has succeeded, updating a state of the successful embedded software service; and

when receiving an indication that invocation of one of the one or more embedded software services was unsuccessful, using the unique ID associated with the unsuccessful embedded software service to re-invoke the unsuccessful embedded software service.

21. (New) The method of claim 20, wherein the invocation map is further configured to store an object holding a context of a shared memory such that changes to the shared memory based on the context-sensitive attribute are reflected.

22. (New) The method of claim 20, wherein when receiving an indication that invocation of one of the one or more embedded software services has succeeded comprises

determining whether all of the embedded software services of the composite software service are consumed successfully including a successful routing of any services that failed.

23. (New) The method of claim 22, wherein when receiving an indication that invocation of all of the one or more embedded software services has succeeded further comprises:

determining whether the composite service is a root service such that it is contained within another composite service;

if the composite service is a root service, committing together all of the one or more embedded software services; and

if the composite service is not a root service, waiting until the composite service receives a commit or rollback call from a context of the another composite service.

24. (New) The method of claim 20, wherein when receiving an indication that invocation of one of the one or more embedded software services was unsuccessful comprises determining whether any of the embedded software services of the composite software service are consumed unsuccessfully without having a predefined routing upon failure.

25. (New) The method of claim 24, wherein when receiving an indication that invocation of one of the one or more embedded software services was unsuccessful, rolling back a context of the instance of the composite software service.

26. (New) The method of claim 20, wherein the invocation map further stores modifications to shared data structures accessed in a definition of the composite service as an object holding a context of the shared memory and updating the invocation map to reflect changes or discarding changes done to the shared memory based on a context of the composite service instance.

27. (New) The method of claim 20, wherein the one or more embedded software services is associated with a service library, wherein invoking one of the one or more embedded

software services comprises using the unique ID associated with the invoked embedded software service to access a wrapper interface associated with the service library.

28. (New) The method of claim 27, further comprising:

obtaining a context of the service library;

preparing a software service associated to the implementing library within the unique context obtained from the library; and

performing a recovery behavior on the library context upon a termination event.

29. (New) In a computing system having one or more computers configured to invoke one or more composite software services, each composite software service containing one or more software services, a method of executing predefined transactional and recovery behavior, the method comprising:

invoking a composite software service containing one or more software services;

if each of the one or more embedded software services in the composite software service is consumed successfully or routed successfully and the composite software service is a root service, issuing a commit call to the one or more embedded software services; and

if each of the one or more embedded software services in the composite software service is not consumed successfully or routed successfully, issuing a rollback call to the one or more embedded software services, wherein the rollback call comprises:

if a flow port on one of the one or more embedded software services is an overwriting port marked for rollback and is connected to another software service, invoking a rollback of the connected software service without invoking a rollback of the underlying service;

if a flow port on one of the one or more embedded software services is an extending port marked for rollback and is connected to another software service, invoking a rollback of the connected software service after completing a rollback of the underlying service; and

if a flow port on one of the one or more embedded software services is a compensation port connected to another software service, invoking a rollback of the connected software.